Automatic recognition and exploitation of patterns in ontologies

By Ondřej Šváb-Zamazal and Vojtěch Svátek

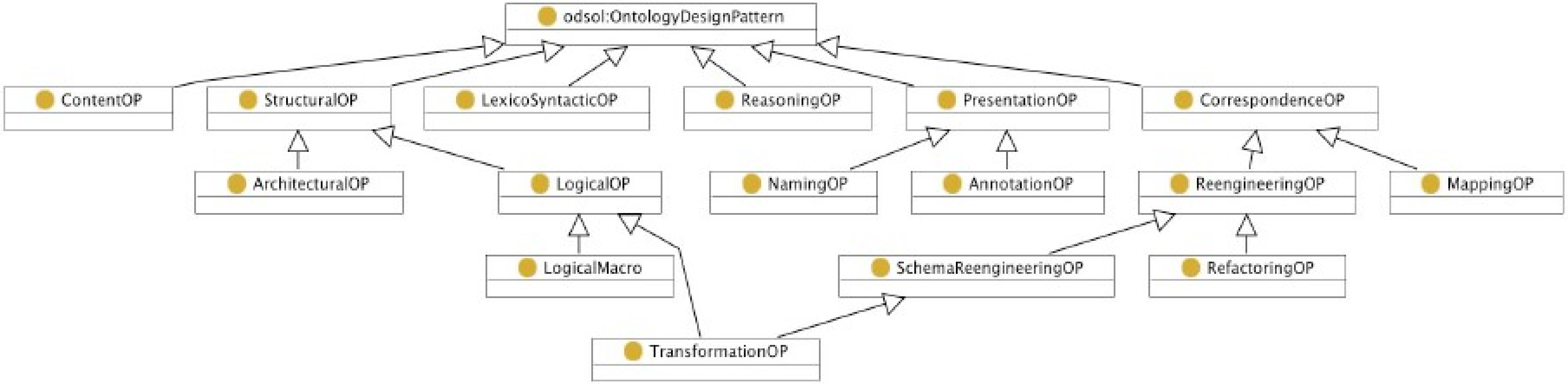
Outline

- Ontology design patterns
- Name-structural patterns in ontologies
- Error mapping patterns in alignments
- Correspondence patterns
- Semantic structures as patterns

Ontology design patterns

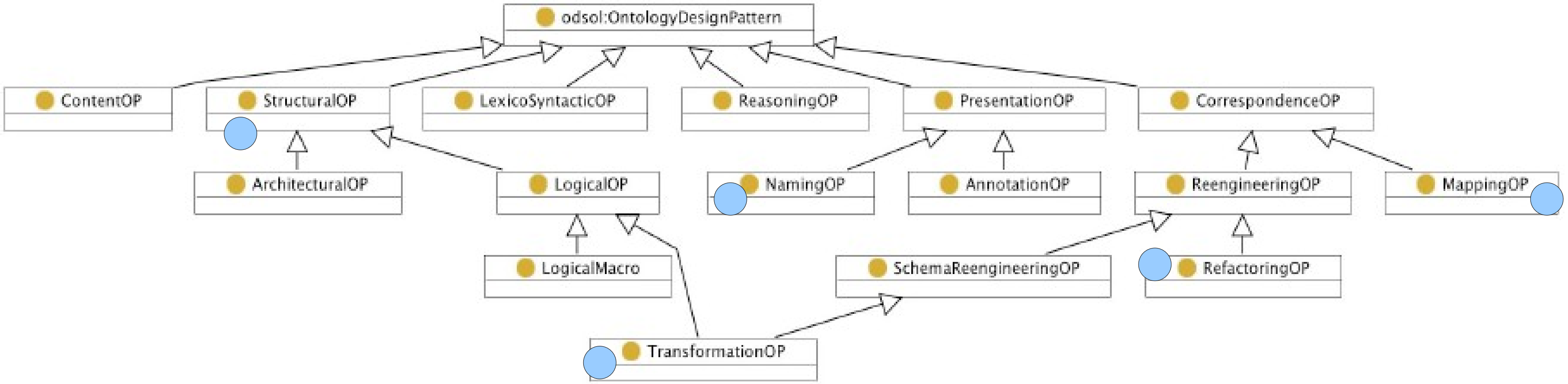
- Another kind of support for ontology development
- Ontology building blocks that allow design by re-engineering, specialization, and composition
  - Logical patterns
  - Content ontology design patterns
  - Ref: http://www.ontologydesignpatterns.org/

Ontology design patterns



Gangemi A.: Ontology Design. In tutorial of Introduction to the Semantic Web.

Ontology design patterns



Gangemi A.: Ontology Design. In tutorial of Introduction to the Semantic Web.

ODP – logical patterns

- Patterns for representation modelling choices in specific language (eg. OWL)
- Logical constructs (of language) and their composition → Language-dependent
- Contains logical vocabulary without specific content → domain-independent

ODP – logical patterns

- Representing ...
  - Classes As Property Values
  - Specified Values
  - Part-whole relations
  - N-ary Relations
  - Roles

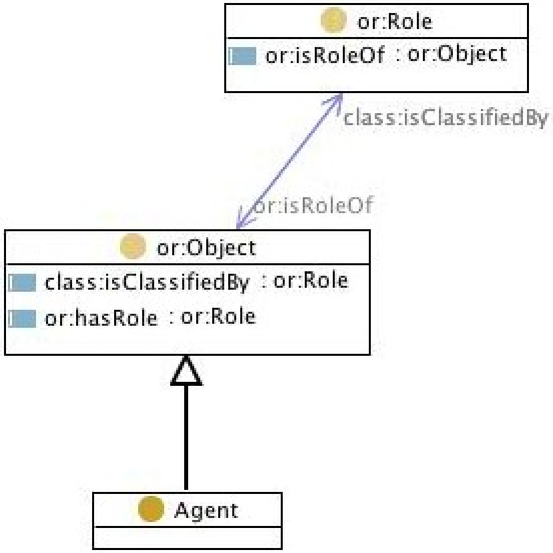
Ref:http://www.w3.org/2001/sw/BestPractices/OEP/

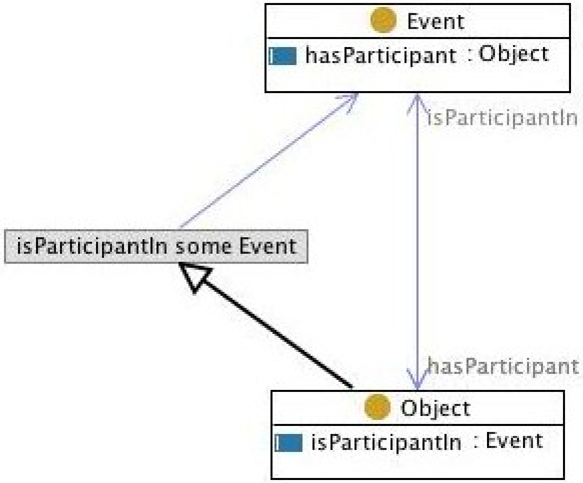
ODP – content patterns

- Patterns for modelling certain kind of situations/cases/problems in ontologies
- Contains non-logical domain specific vocabulary → domain-dependent
- Language-independent
- Best practices in a domain
- Takes advantage of Upper-level ontologies

ODP – examples of content patterns

- *Agent role*: to represents agents and the roles they play
- *Participation*: to represent the participation of objects in events





Taken from ODP portal: http://ontologydesignpatterns.org/

Name-structural patterns in ontologies

- Assumption:
  - Designers can benefit from self-explaining entity names (URIs)
  - Entity naming reflects the set-theoretic meaning of these entities
- Name patterns
  - Captures relation between entity names and the position of those entities in the ontology structure (logical axioms)

Name-structural patterns in ontologies (contd.)

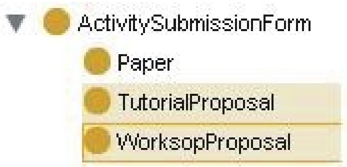
- Violation of name patterns due to:
  - Failure to properly identify the set-theoretic semantics
    - eg. ProgamCommittee, subClassOf(ProgramCommittee)=CommiteeMember
  - Bad naming policy
    - eg. Author, subClassOf(Author)=Scientific
  - Use of synonymy or hyperonymy
    - eg. Presentation, subClassOf(Presentation)=InvitedTalk

Non-matching child (I)

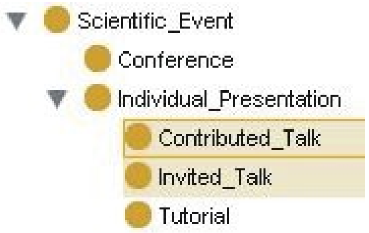
- Simple subsumption violation
- Examples:
  - – Car, subClassOf(Car)=Wheel
  - – Paper, subClassOf(Paper)=Accepted
- Higher precision with thesaurus
- Use case: evaluation of ontologies

Matching siblings with non-matching parent (II)

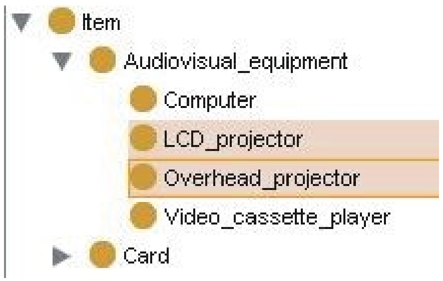
- refinement of the simple subclass pattern
- less frequent
- possible indicator of overly flat hierarchy?



ActivitySubmissionForm
- Paper
- TutorialProposal
- WorksopProposal



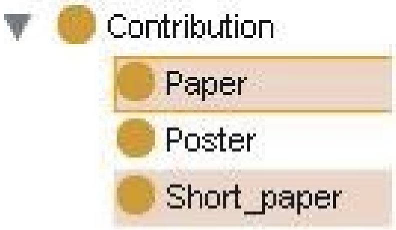
Scientific_Event
- Conference
- Individual_Presentation
  - Contributed_Talk
  - Invited_Talk
  - Tutorial



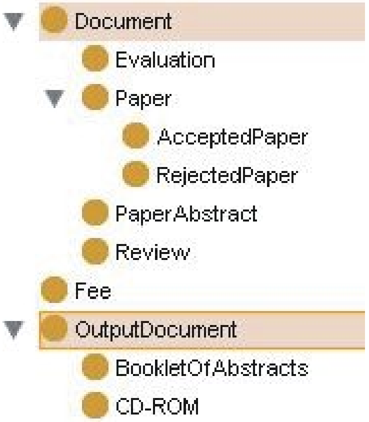
Item
- Audiovisual_equipment
  - Computer
  - LCD_projector
  - Overhead_projector
  - Video_cassette_player
- Card

Matching siblings with non-matching parent – *variant* (III)

- special case of previous pattern where one of the siblings has a single-token name



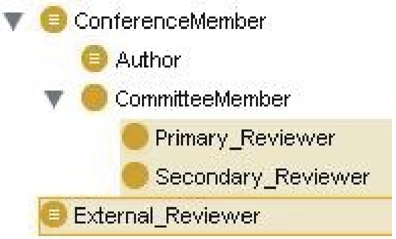
- Contribution
  - Paper
  - Poster
  - Short_paper



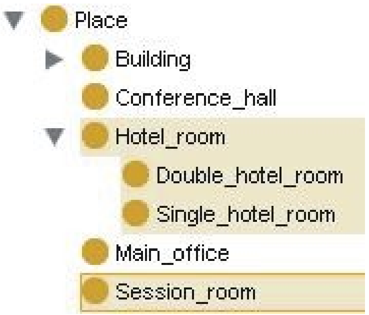
- Document
  - Evaluation
  - Paper
    - AcceptedPaper
    - RejectedPaper
  - PaperAbstract
  - Review
- Fee
- OutputDocument
  - BookletOfAbstracts
  - CD-ROM

Matching outlier (IV)

- Possibly a disconnected structure of entities of same type (e.g. effect of uncoordinated updates)
- May also be polysemy/homonymy



- ConferenceMember
  - Author
  - CommitteeMember
    - Primary_Reviewer
    - Secondary_Reviewer
- External_Reviewer



- Place
  - Building
  - Conference_hall
  - Hotel_room
    - Double_hotel_room
    - Single_hotel_room
  - Main_office
  - Session_room

Matching property with relata

- Property-oriented pattern
- Also pattern names could be self-explanatory
  - eg. 'has(Person, Car)' vs. 'owns(Person,Car)' or 'owns-car(Person,Car)'

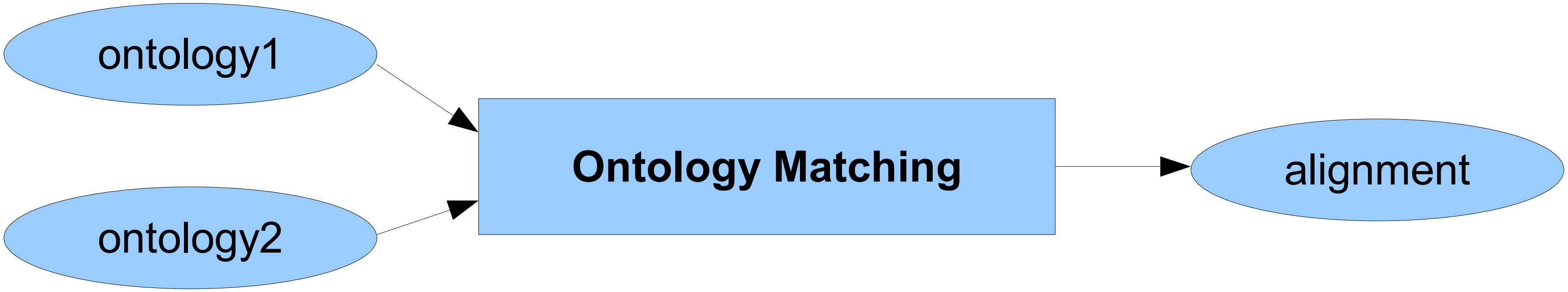
- *Not represented and implemented yet*

Matching subproperty relata with property relata

- domain/range of subproperties should correspond with domain/range of properties
- Example:
  - domain(writtenBy)=Document, range(writtenBy)=Person
  - domain(reviewWrittenBy)=Review, range(reviewWrittenBy)=Possible_Reviewever

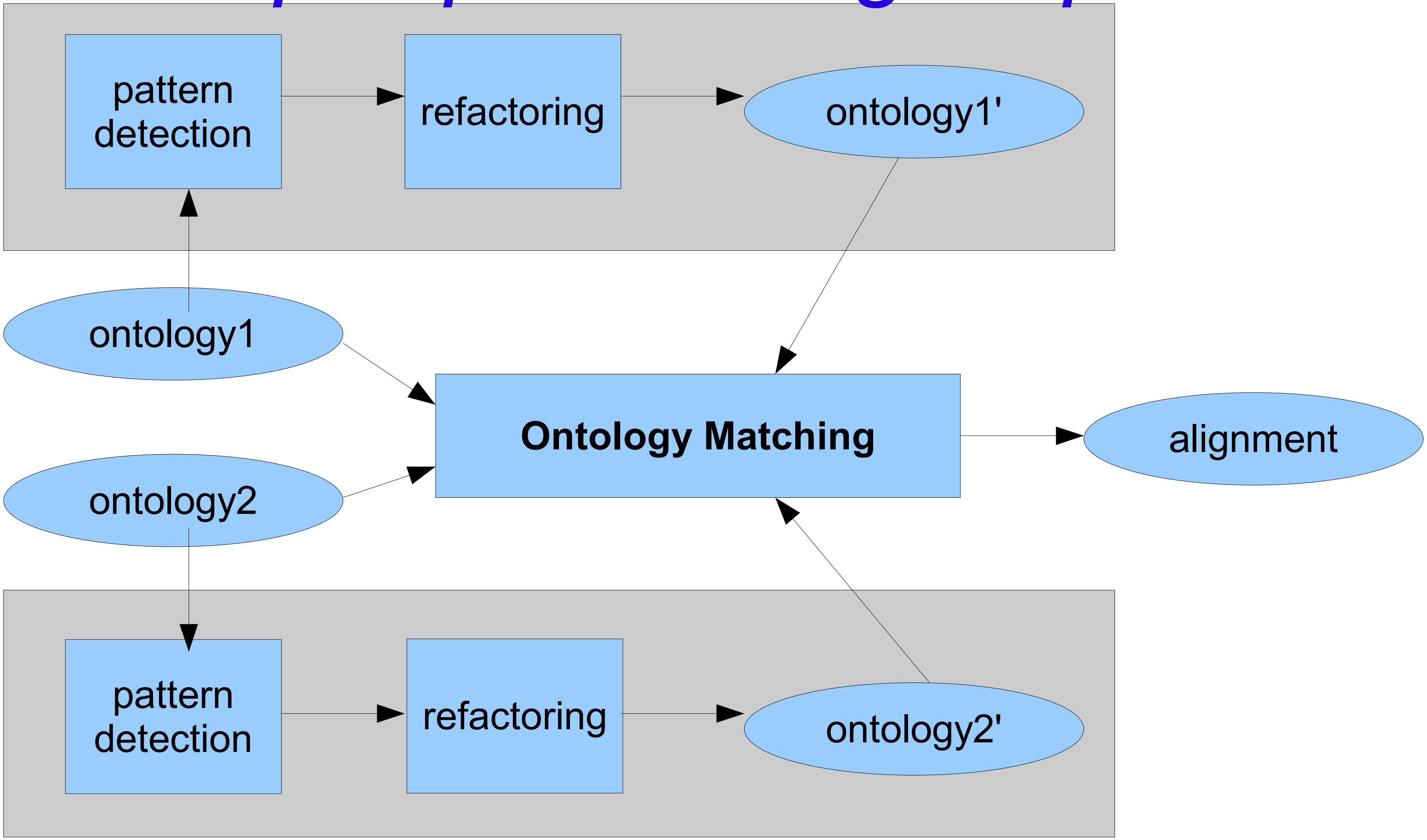
- *Reffered to as RBox compatibility*
- *Not represented and implemented yet*

Ontology Matching driven by pattern detection



Use Case

OM driven by pattern detection *pre-processing step*



OM driven by pattern detection *pre-processing step*

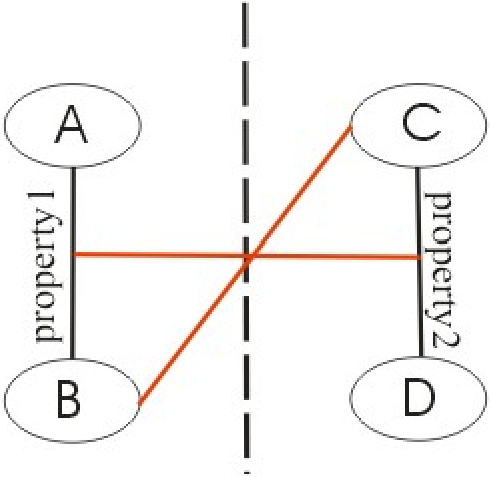
- Refactoring:
  - Three semantic-preserving operations: 'rename', 'add', and 'restructure'
  - current work in progress: automatic refactoring
- Pattern detection:
  - Discovery of head noun improved
  - two variants of implementation: OWL API, SPARQL queries
  - Future work:
    - poor handling of multiple inheritance
    - term synonymy and polysemy

Mapping Patterns in alignments

- Patterns dealing with (at least) two ontologies
- Reflect the structure of ontologies and include correspondences between elements of ontologies
- Kinds of Mapping patterns:
  - For improving incorrect correspondences – error mapping patterns
  - For designing 'smarter'/complex correspondences – correspondence patterns

Error mapping patterns - *domain-range mismatch*

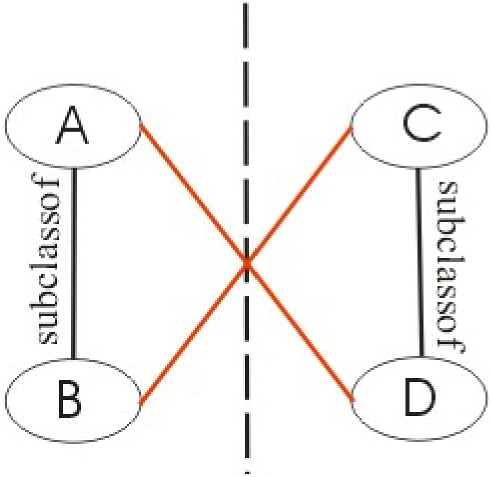
- eq. correspondences between 2 classes and eq. Correspondences between 2 properties
- properties with the same relata exluded
- Incorrect propert-to-property correspondence: inverse properties
- Incorrect class-to-class correspondence



Example (conference.owl and paperdyne.owl)*:*
Reviewed_contribution=Reviewer
reviews=reviews

Error mapping patterns - *criss-cross mismatch*

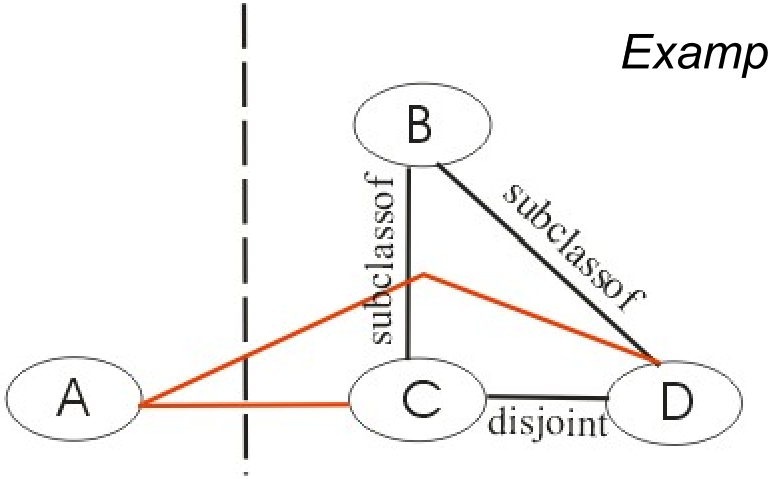
- eq. correspondences between children and parents



Example (crs_rd.owl and iasted.owl):
article=Item
document=Document

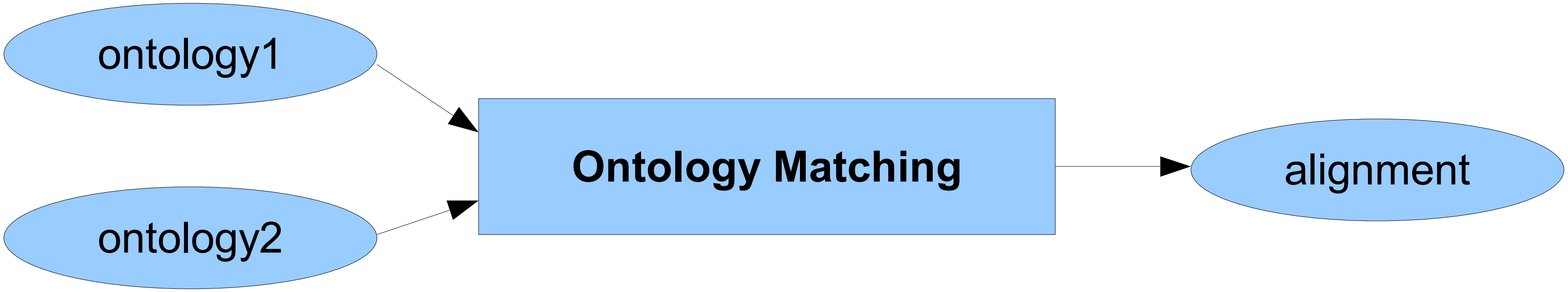
Error mapping patterns - *disjoint siblings mismatch*

- eq. correspondences with disjoint sibling classes



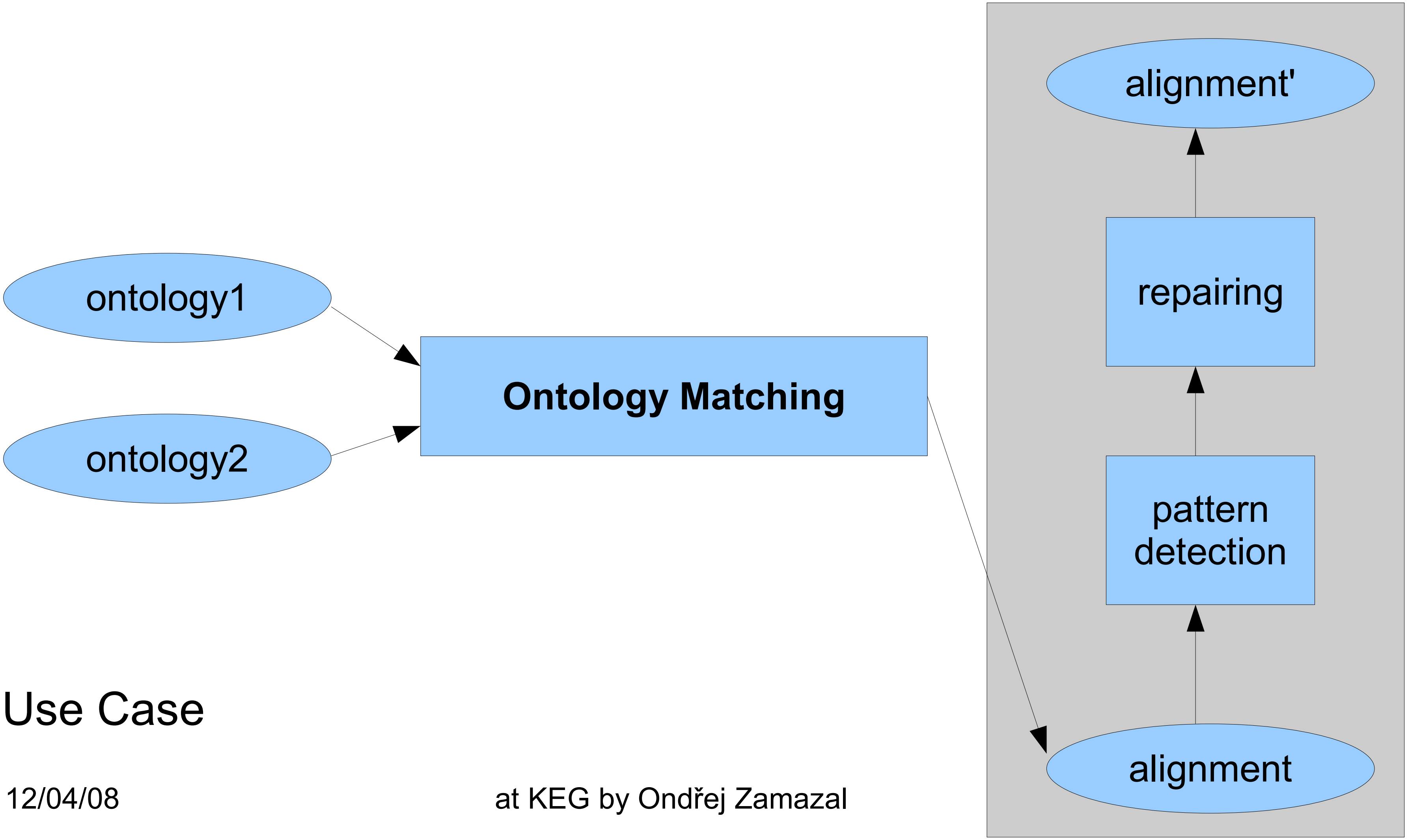
Example (opencof.owl and conference.owl):
Committees=Organizing_committee
Committees=Steering_committee

Ontology Matching driven by pattern detection



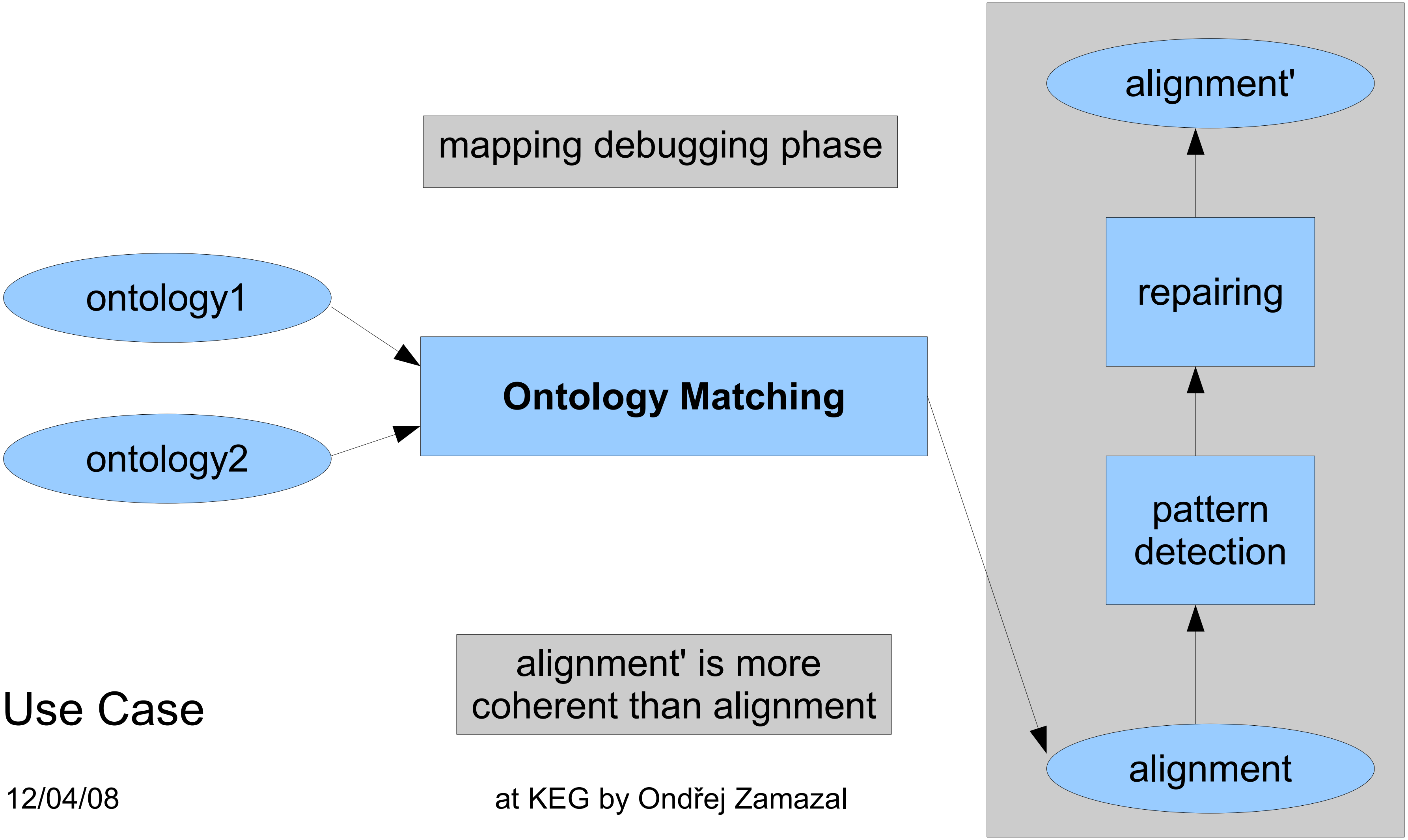
Use Case

OM driven by pattern detection *post-processing step*



Use Case

OM driven by pattern detection *post-processing step*



Use Case

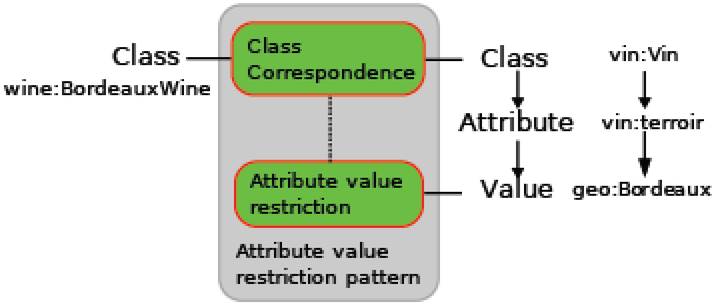
Correspondence patterns for design

- Correspondence patterns by Francois Scharffe
  - As smarter mapping between ontologies
- Current algorithms discover simple and error-prone correspondences → need for user involvement
- CPs are primarily intented to support the user when creating/modelling complex correspondences
- Pattern template: name, problem, solution, consequences + grounding part of pattern

Ref: http://www.omwg.org/TR/d7/patterns-library/

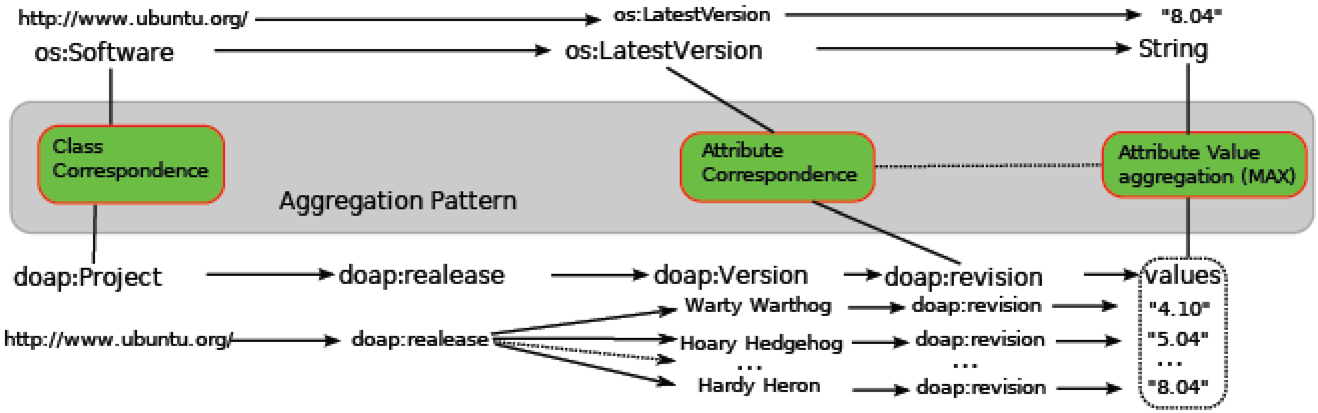
CP – Class by Attribute Value Correspondence

- concept-to-concept correspondence where the class in one ontology is restricted to only those instances having a particular value for a given attribute/relation



CP – Aggregation pattern

- class-to-class eq. correspondence and property-to-property eq. correspondence with aggregation function



Semantic structures as patterns

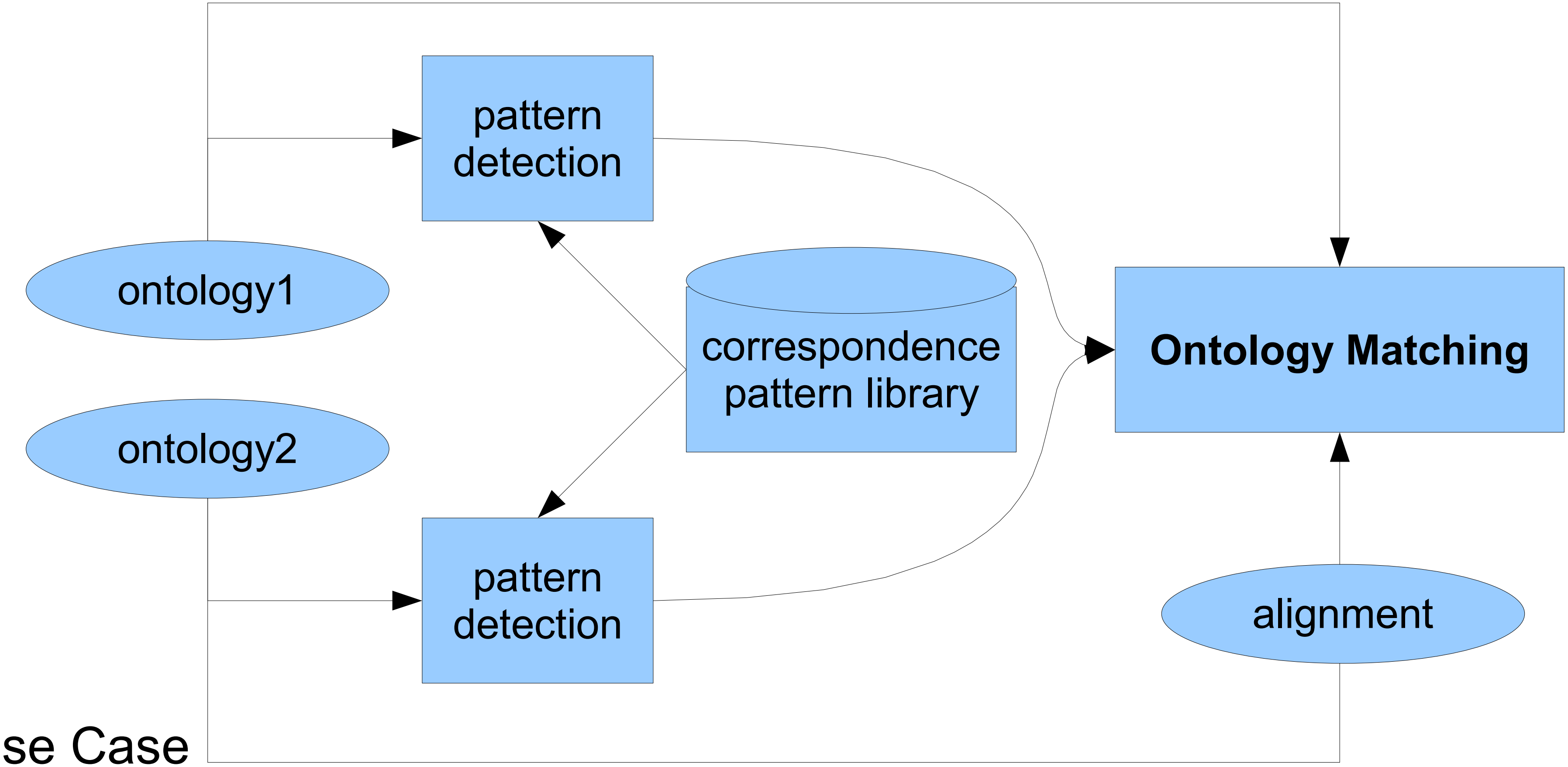
- At the phase of design:
  - problem space ↔ solution space
- Problem space: domain/task modelling problem
- Solution space: modelling choices
- 1 modelling problem : many modelling choices/alternatives
- Instance of modelling choice: semantic structure

Semantic structures as patterns

- Task for ontology matching:
  - Operational aspect: OM tool must consider whole structures and not isolated entities → pattern-based OM with Ontology alignments between diverse modelling choices as output
  - Representation aspect: Simple correspondences do not work → correspondence patterns

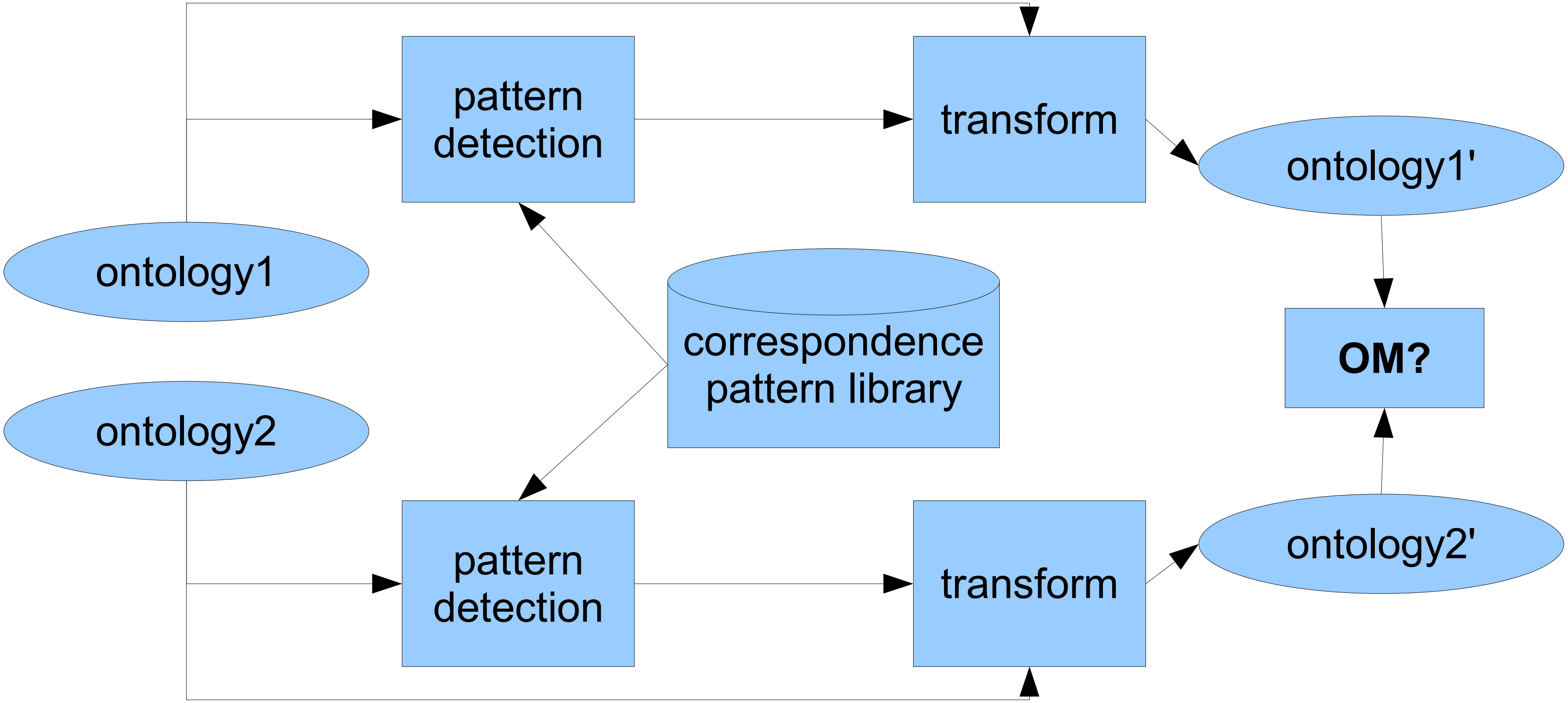
Use-Case: ontology transformation from one modelling choice to another

Pattern-based OM



Use Case

Transformation of ontologies

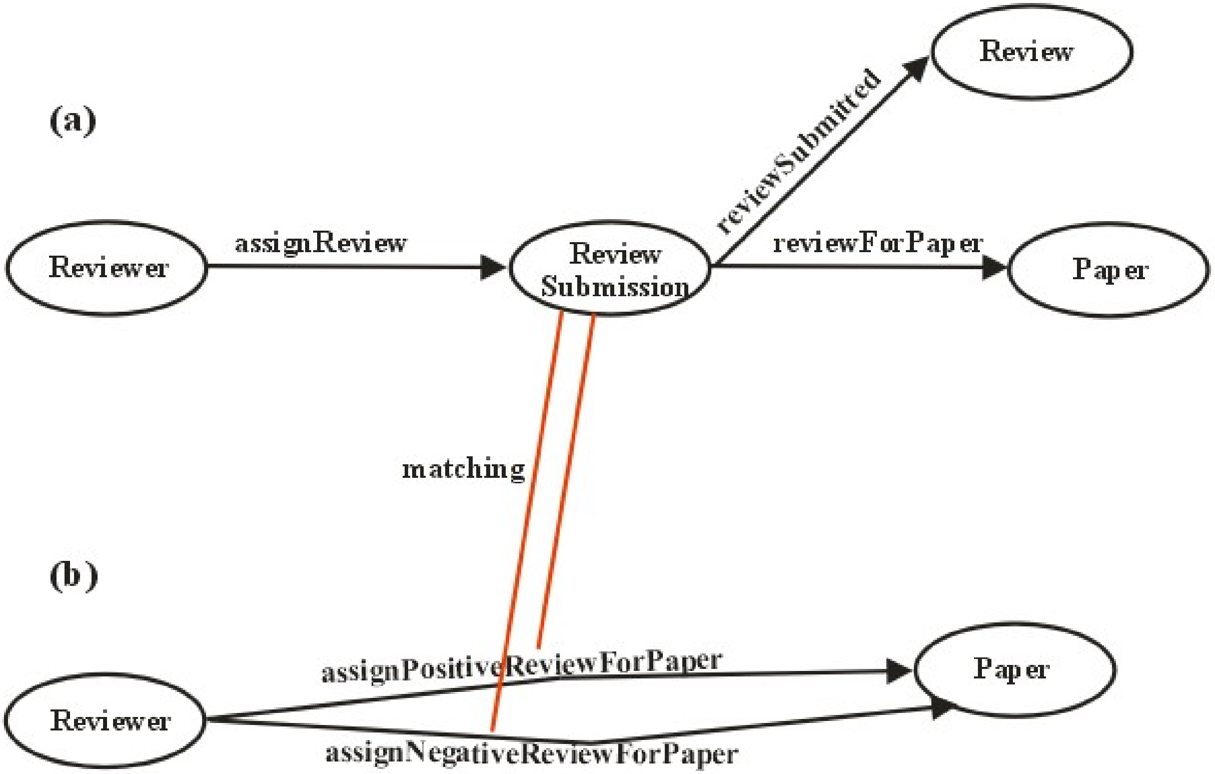


Use Case

Semantic structures as patterns - N-ary relations in OWL

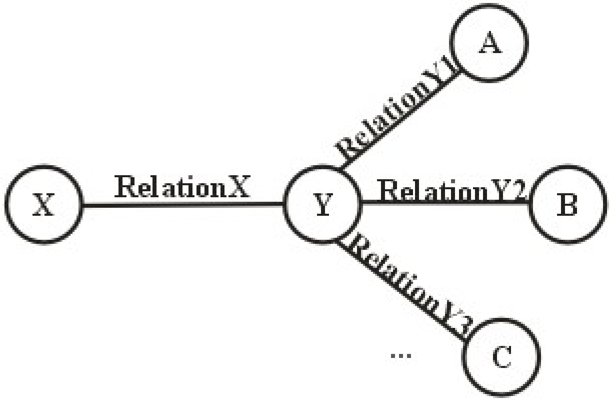
Modelling alternatives:

- (a) Reify the whole relation
- (b) Use distinct binary relations



Pattern for N-ary Relation Discovery

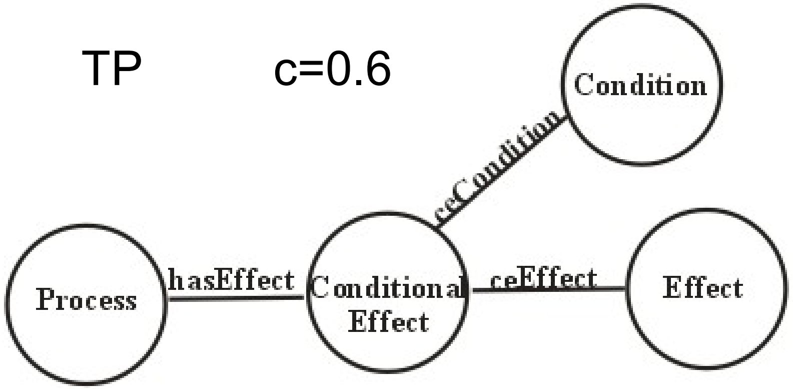
- Structural pattern:

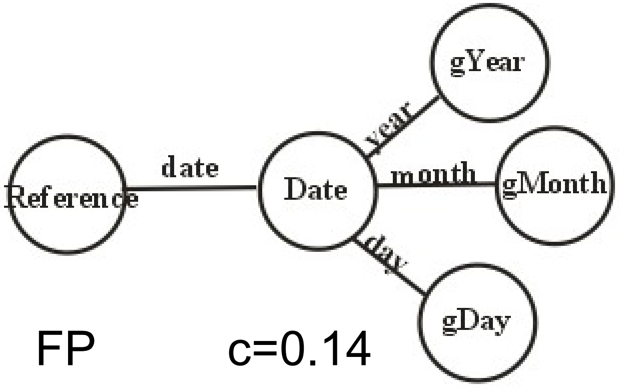


- Naming pattern:
  - The average token-based similarity measure *c* between 'RelationX' and other entities from structural bunch

Example of detection of N-ary relation

TP c=0.6

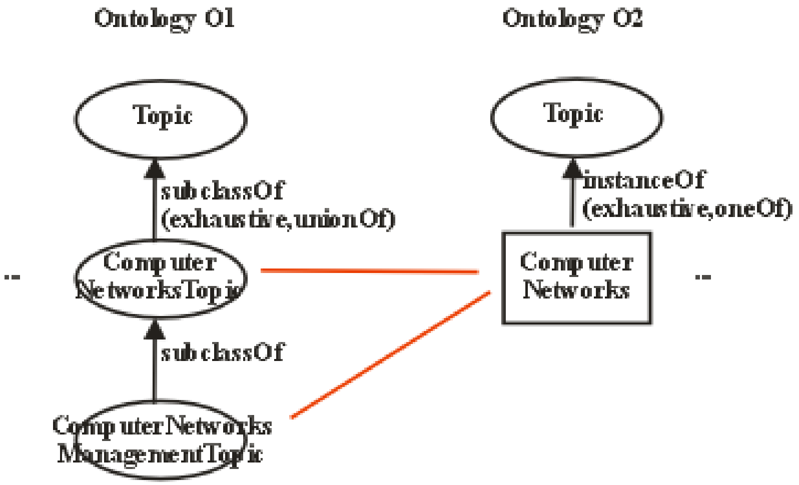




FP c=0.14

Semantic structures as patterns - Value partitions in OWL

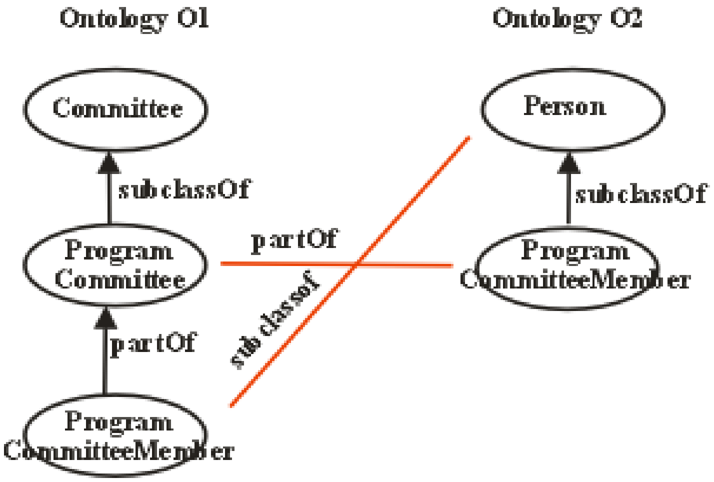
- Modelling alternatives how to represent specified collections of values (qualities, attributes, features):



currently no pattern

Semantic structures as patterns - Value partitions in OWL

- Modelling alternatives how to represent part-whole relations:



currently no pattern

Conclusions & Future work

- Various ontology patterns (not only) useful in different phases of ontology matching
- Future work related to different patterns:
  - Name-structural patterns:
    - Automatic refactoring, multiple inheritance, thesaurus
    - More systematic pattern creation
  - Error mapping patterns:
    - Repairing and evaluation of this phase

Future work

– Semantic structures:
  - Capturing them as detectable patterns
  - Systematic Extension of current pattern stock with potentially usable diverse ontology patterns (eg. parts of correspondence patterns etc.)